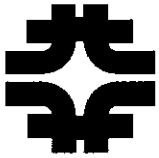

Fermi National Accelerator Laboratory

FERMILAB-Pub-93/006

Loosely-Coupled Parallel Processing at Fermilab

Frank Rinaldo and Stephen Wolbers

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

January 1993

Submitted to *Computers in Physics*

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CHO3000 with the United States Department of Energy

Disclaimer

Loosely-Coupled Parallel Processing at Fermilab

Frank Rinaldo and Stephen Wolbers

Fermilab, Batavia IL 60510 USA

Introduction

Progress in high-energy physics (HEP) has depended on increasing the availability of computing power and exploiting it in new ways. Many different techniques have been developed to increase computing power. One recent approach has involved the use of dedicated clusters of RISC-based workstations. Such clusters provide large increases in computing power at an affordable price. Parallel processing software has been developed to take advantage of the clusters, allowing groups of workstations to operate as single computing engines.

In this article we will provide an overview of HEP computing, emphasizing the offline event-reconstruction process that requires large amounts of computing. We will discuss Fermilab's software package for parallel computing, and describe the Fermilab farm clusters and experience with parallel computing. Finally, we will describe additional computing initiatives that promise enhanced computing environments as well as possible future directions of research.

Offline Computing

Fermilab is a national laboratory dedicated to HEP research. Experiments are conducted both at the proton-antiproton collider, which at 1.8 TeV in the center of mass, is the highest-energy collider in the world, and in the fixed-target areas, where extracted proton and secondary beams are used in many different elementary-particle investigations. All aspects of HEP require computing. On the experimental side, computing is intensively used for detector and accelerator design, data acquisition and accelerator operations, data reconstruction, data analysis and document preparation. Theoretical

HEP also utilizes increasing amounts of computing for many calculations, and lattice QCD requires extremely large amounts of computing. This article will concentrate on one aspect of HEP computing: offline event reconstruction. In order facilitate understanding of offline event reconstruction, we will outline what is needed to collect and analyze data in an experiment.

A detector consists of many active elements. The signals on these elements are read out when an interesting interaction (an 'event') occurs during the collision of very-high-energy particles. An online data-acquisition system collects the information, and writes it to a storage medium (normally magnetic tape) for further analysis at a later time. A series of algorithms are used to convert the information on these data tapes into quantities that can be used during further analysis. These programs are usually referred to as 'event reconstruction programs'. The programs are written in Fortran, can be quite large (over one million source lines) and are maintained and modified over many years.

Output from event reconstruction consists of the trajectories and physical parameters of the particles in the event (energy, momentum, charge, etc.) as well as information used in further analysis of the data. The next step in the analysis is the splitting of the data into subsamples, each consisting of all the events that are interesting for an analysis topic. These events are written onto special sets of files (tape or disk), sometimes in a compressed format. Finally, the physicists involved in the experiment write their own programs to read and analyze these datasets and to produce the final publishable physics results.

The offline-reconstruction codes are quite large, and reconstructing an entire experiment's data set requires a huge amount of computing time (CPU time). The total CPU time needed is simply the number of events collected multiplied by the average CPU time needed to reconstruct each event. The need is growing rapidly as the detectors become more complex, requiring more CPU time per event, and as the number of events collected by each experiment increases. Our unit for expressing CPU time is the "MIP-'time' (Million Instructions/sec)," which is measured using a suite of standard

HEP codes, or benchmarks, and where 'time' is seconds, minutes, hours or any other convenient measure. We define the performance of this set of benchmarks on a VAX 11/780 to be equal to 1.0 MIPS.

HEP computations are quite large. Individual event sizes written to tape range from an average 3 kbytes for small detectors up to 400 kbytes for the largest collider detectors. The number of events collected by an individual experiment range from 10 million up to 20 billion. Total event sample sizes on tape for each experiment vary from 2 to 40 Tbytes.

Offline computing needs for a single run of an experiment typically range from 50 MIP-years to over 5,000 MIP-years. The typical experiment today requires a few hundred MIP-years. These requirements are so large, and are growing so rapidly, that standard computing techniques cannot provide a computing solution within available time and budget constraints. Industry solutions tend to lag behind these computing needs. This problem has been recognized for many years as one that can be solved with parallel-processing techniques [1]. The individual events are independent of each other. They can therefore can be sent (farmed out) to individual processes, each of which runs a copy of the full reconstruction program. This is the standard 'farm' approach to parallel computing.

Fermilab has been involved in finding solutions to large computing problems using distributed computing for quite some time. The original ACP (Advanced Computing Project) farms utilized Fermilab-built CPU boards (68020-based ACP nodes) communicating over VME with a host microVax [2]. Software was written to distribute events and calibration constants across the nodes and coordinate reconstruction on the nodes. This solution was cost-effective and extremely successful in solving large reconstruction problems.

The advent of RISC processors led to the next phase of development in loosely coupled systems. The MIPS R3000 CPU running UNIX was used as the basis for ACP II, a system that provided a huge increase in CPU power. A new software package, CPS (Cooperative Software Processes) was written to provide the parallel processing on these

systems. Events in the marketplace dictated the purchase of Unix workstations as a cost-effective technique for providing UNIX computing. The increase in CPU power, and decrease in cost, of the UNIX workstation have made it an extremely attractive platform for this application. The Fermilab UNIX farms grew out of the concept of coupling many such workstations in a dedicated computing resource.

Cooperative Processes Software

The software that enable parallel processing to be carried out is an important component of the workstation farms. This software must connect the workstations in a way that is transparent to the developers of the large computing programs that run on these machines. CPS, [3,4,5] the parallel programming toolkit developed at Fermilab, provides a cost-effective solution to the enormous computing requirements in experimental HEP. CPS programs run as a collection of processes distributed over many computers. Typically, these computers are high-performance UNIX-based RISC workstations, which are networked together via Ethernet, dedicated to the task of acting as a distributed-memory supercomputer, and not used individually as workstations. This mode of computing is very effective for jobs that have relatively low I/O requirements in comparison to CPU time (i.e. 500-5000 machine instructions per byte of I/O).

Parallel programming with CPS is often described as 'large-grained' and 'loosely coupled'. Large-grained means that the processes running in parallel are typically entire subroutines or programs. The term loosely coupled describes a set of processors running relatively independently of each other on a network, where communications are slower than in more tightly coupled processors. Our current needs at Fermilab are satisfied using Ethernet at 10Mbits/sec as the means of connecting the nodes (RISC workstations) together.

CPS supports several models of parallel programming. The client/server model is supported with remote subroutine calls. Processes may also interact via message passing, process synchronization, bulk data transfers, and a mechanism called process queues. The remote-subroutine-calls mechanism allows a computer program to make

asynchronous subroutine calls in which the subroutines can reside on other computers. This feature allows a job to distribute work across many computers.

Message passing is a generic mechanism that supports distributed computing. Any process can communicate with any other process on any machine. Messages may consist of data or instructions (flow control). Process synchronization is a mechanism whereby various processes (under user control) can synchronize. Once work has been distributed over many computers, there is usually a need to 'collect' and process the results that has been completed in parallel. Work can again be distributed as needed. Bulk data transfers allows large 'blocks' (or amounts) of data to be moved between processes in a fast and efficient manner. Usually, small chucks of data are transferred via message passing, but the choice remains with the programmer. Process queues are a powerful mechanism allowing processes to place themselves (or even other processes) into a user-defined queue. Worker or analysis processes can place themselves into a user-defined idle queue. Another process can remove an idle process from this queue, assign it some work, and keep distributing work until the queue becomes empty (or there is no more work to process). The mechanism is self-balancing, that is, if a particular worker process completes it's task quickly, it will place itself back on to the idle queue (i.e 'fast' worker processes will enter the idle queue more often than 'slow' worker processes and thereby process more work).

CPS allows the user complete control in structuring their CPU resources effectively. Figure 1(a) shows the most common way in which CPS jobs are structured for HEP event reconstruction. Figs. 1(b) and 1(c) show other examples of how a CPS job might be structured. CPS currently runs on a number of UNIX-based workstations (Silicon Graphics, IBM, DEC, HP and SUN) and supports a heterogeneous computing environment. A CPS 'job' could be run on a mixture of any of the previously mentioned systems, or on all of them. Fermilab currently supports CPS for both Fortran and C applications.

A number of other tools that support CPS have also been developed. CPS-Batch is a generic UNIX batch system for submitting and queueing jobs to the farms. This

tool also supports an X-Window operator interface (Fig. 2), tape handling facilities, communications with operations and the ability to reconfigure farm production systems. A farm production system for HEP reconstruction typically consists of an I/O node (with an appropriate number of tape and disk drives) and a number of worker (or analysis) nodes. Allocation of farm resources are reviewed each week, and determined by dynamic priorities set by the Fermilab physics program (but rarely changed more often than on a weekly basis). Analysis tools help to determine the optimal number of worker nodes for a particular experiment and a particular set of code. To make effective use of the CPU resources of the worker nodes, each of these nodes usually runs two or more analysis processes. The actual number of worker nodes assigned to a job, again, depends on the ratio of I/O to CPU needs. Jobs that are highly CPU bound might be assigned 15-25 worker nodes per production system and having several production systems assigned to them.

Performance Experience

Since early 1991, CPS and CPS-Batch have been in use for production (7 day/week, 24 hours per day). The Fermilab UNIX farms consist of approximately 100 Silicon Graphics (SGI) workstations (models 4D/25 and 4D/35) and approximately 100 IBM RS6000 workstations (models 320 and 320H) used as worker nodes (Figure 3). Each node contains a local disk for system control, paging and swapping, and is equipped with an adequate amount of memory for HEP codes (typically 16 MBytes). The farms are in the process of being expanded with the addition of 80 SGI workstations (IRIS Indigo) and 44 IBM workstations (model 220). The total computing power available is approximately 4500 MIPS (soon to be 8500 MIPS).

In addition to the worker nodes mentioned above, a separate set of machines (I/O nodes) have been acquired to provide the connectivity to tape and disk. There are 6 IBM RS6000's (models 320 and 530) and 3 SGI workstations (2-processor model 4D/420) for this purpose. Tape and disk are connected via SCSI to these nodes. A total of 70 8-mm tapedrives and 60 GBytes of disk storage are spread across all the systems. Ethernet

is used to connect the machines, and the farms are segmented, with approximately 15 worker nodes and one I/O node per segment. In some cases, I/O nodes contain multiple Ethernet interfaces, and thus are placed on multiple segments. This arrangement allows the most efficient movement of data from I/O to worker nodes (Figure 4). Between three and six 8-mm tapedrives and 2-8 GBytes of disk are attached to each such subsystem. The Network File System (NFS) is used to provide access to the disk across the entire subsystem.

Many experiments at Fermilab either have run or are running jobs on the UNIX farms. They include E665, E687, E706, E731, E760, E771, E789, E791, D0 and CDF. CPS-Batch utilizes a concept, called production systems, to place processes on the proper hardware subset. This capability allows allocation of tape drives, CPU, disk space, etc. to match the needs of the physics program and the user group. A group in full production may run on three or more production systems simultaneously.

The 10 groups that have converted their code to CPS have all been able to make the conversion easily. Modifications to user code are fairly small, and are usually accomplished in a matter of weeks (including testing and debugging). For some users, CPS itself has had certain features modified to satisfy varying performance needs. CPS-Batch has been the major weakness in overall system performance. A great deal of effort has gone into making the batch system more robust and providing easier recovery from the multiple failures inherent in this type of distributed hardware. Even so, users have been encouraged to checkpoint long jobs, at intervals of approximately 1 h, so that failures only cause 1 h of lost computation. Some jobs run as long as 6 or 7 days.

A very important aspect of operation is the communication among various groups using and supporting the UNIX farms. Three major methods that have been used to provide this communication. First, there are weekly meetings to which all users and support personnel are invited. Second, a mail list has been established to which all users and support people can send requests, complaints, or other information. Third, and most importantly, users can call the computer operators 24 hours a day to report

problems, and support personnel can be paged at all times to correct problems. The combination of the three communication channels has been crucial in providing reliable service.

The ratio of CPU power to I/O transfer rate is an important consideration in any parallel-processing application. The more CPU-intensive a job is (per I/O transfer), the easier it is to gain increased computing in a parallel system. An ideal situation is one in which each processor is busy 100% of the time (corresponding to 100% efficiency), and each additional node adds exactly that node's computing power to the problem (linear speedup).

In general, it is not easy to predict how well a CPS job will utilize the resources of its production system. Many factors influence the total throughput. We have relied on empirical determination to allocate production systems to users. In the best cases (Monte Carlo or offline codes transferring very large blocks of data) the speedup is linear up to more than 15 nodes, and the efficiency is above 90%.

Arrival at the current set of configurations of the farms was not simple. Various attempts were made during the installation of systems to connect farms and subfarms in configurations which, unfortunately, could not support robust production. Problems included node crashes, hangs, SCSI resets, and other assorted 'glitches'. Due to the production nature of the systems, such glitches were not acceptable, and significant effort went into reconfiguring and modifying operations until steady running conditions were achieved. Though not glitch-free, the current farms are rather stable and reliable.

The total amount of computing time used in event reconstruction has steadily increased as we have added nodes to the farms and as we have improved the CPS and CPS-Batch software. As much as 2300 MIP-months/month have been delivered to user applications (see Fig 5). Three experiments have completed reconstruction efforts on the farms, and one of them (experiment E760) reconstructed over one billion events using only a small fraction of the farms (very efficiently). The large collider experiments are able to keep up with the maximum data-taking rate in the 1992-93 data run.

Other Efforts; Possible Futures

Many computing problems are being addressed by Fermilab. The biggest problem (in CPU terms) are event reconstruction and Monte Carlo simulation, and users have discovered that the clustered-workstation approach solves these problems efficiently and inexpensively. However, increases in CPU power per processor are expected to continue, but not to be matched by a corresponding growth in communication speed. Thus, a bottleneck in communications will eventually hamper the effectiveness of the cluster solution.

Fermilab is also working on tightly coupled parallel computing to solve grid-based problems HEP problems such as Lattice quantum chromodynamics (QCD). A very large parallel computer built at Fermilab, ACPMAPS [6], consists of tightly coupled Intel i860 processors. ACPMAPS has a peak speed of 50 GFlops and low latency for small message transfers. A software package, CANOPY [7], allows scientists to cast their parallel-processing problems in natural terms without worrying about the underlying hardware. This approach has been very successful in producing physics results in Lattice QCD.

The knowledge that we have gained in running these systems in real production environments enables us to move on to other computing problems. Many HEP problems, and computational efforts in other scientific endeavors, involve access to large datasets by many researchers. These problems and efforts involve much smaller compute-to-IO ratios than the those in event reconstruction, and invoke new issues of data storage and retrieval. Fermilab is working to provide solutions in this area. CLUBS (Clustered Large UNIX Batch System) is one project that require access to large quantities of data, as well as large amounts of computing. Initial configurations uses UNIX workstations as compute engines coupled via high-speed connection (Ultranet) to an Amdahl mainframe for data access. Two goals of the project are to replace the Amdahl with a completely UNIX-based data-serving system, and to provide access to large amounts of data. Other areas of research at Fermilab are aimed at providing data and computing to end users, and include such diverse areas as Vax Clusters for data serving, general UNIX computing

for a large user community, and special parallel computing hardware for data serving and analysis.

Parallel programming has provided a solution to many large computing problems. Special software packages attempt to simplify the work needed to take advantage of parallel processing, both in the loosely coupled case, allowing exploitation of cheap UNIX workstations, and in the high-performance, tightly coupled environment of massively-parallel computers. Fermilab has successfully provided solutions to both sets of problems with CPS and CANOPY. Production parallel computing is a fact of life at Fermilab.

Ease of programming, debugging, monitoring, administration, allocation, resource management, load leveling, etc. are all in need of improvement. Porting code to parallel systems is never easy. Possible solutions may involve some synthesis of current ideas and packages into standardized tools or parallelizing compilers.

More powerful tools for utilizing parallel processing can only improve the productivity of the people who use the computers. For the present, the rapid advances in microprocessors promise further increases in computing power at falling prices. We can hope that communications will also see rapid gains. Overall, the future looks good for parallel computing in HEP.

Acknowledgments

We wish to acknowledge the work of the Fermilab Groups responsible for the success of the UNIX farms: Matt Fausey, Bill Meyer, David Potter, Marilyn Schweitzer, Kevin Sullivan, Roberto Ullfig and Robert Yeager of the Farms Systems Group and Lisa Amedeo, David Fagan, Marc Mengel, David Oyler and Matt Wicks of the UNIX Systems Support Group. Many valuable contributions have come from Gerry Bellendir, Joel Butler, Peter Cooper, Chuck DeBaun, Mike Diesburg, Paul Lebrun, Mark Leininger, Jeff Mack, Jim Meadows, Frank Nagy, Tom Nash, Al Thomas, Brian Troemel, and Eric Wicklund. The staff of Data Center Services have made tremendous contributions in

keeping the system running and providing valuable suggestions for improvements. Special acknowledgment goes to Chip Kaliher who made production farming at Fermilab work in the first place.

References

[1] T. Nash, Comp. Phys. Comm. 57 (1989) 47.

[2] I. Gaines and T. Nash, 'Use of New Computer Technologies in Elementary Particle Physics' Ann.Rev.Nuc.Part.Sci. 37:177 (1987)

[3] M. Fausey, "CPS and the Fermilab Farms", FERMILAB-Conf-92/163, 1992.

[4] J. Biel, et.al., ACP Cooperative Processes User's Manual ; November 11, 1990, Fermilab Computing Division Library GA006

[5] M. Fausey, et.al., CPS and CPS Batch Reference Guide, CPS Version 2.7.0 ; June 20, 1992, Fermilab Computing Division Library GA008

[6] M. Fischler, "The ACPMAPS System – A Detailed Overview", FERMILAB-TM-1780, 1992.

[7] Details of CANOPY can be found in the CANOPY 5.0 Manual, M. Fischler, G. Hockney, P. Mackenzie.

Figures

[1] CPS supports a variets of process topologies: (a) typical high-energy physics topology; b) complicated interconnection topology (fairly easily coded in CPS); c) symmetric process topology with all processes communicating to each other.

[2] X-Window based tool monitors system status. The large bar at the top becomes bright red when a tape mount is requested by one of the farm systems. Clicking on icons (also highlighted in red) reveals the name of the tape, the tape drive on which it is to be mounted, and whether the tape is to be write-protected or not. Additional information about the UNIX farms is given in the windows at the bottom of the display.

[3] Rack holds 100 IBM RS6000 model 320 and 320H workstations, that function as worker nodes in a part of the Fermilab UNIX farms. The workstations are connected

to a console concentrator system for maintenance and management, and to Ethernet for process intercommunication.

[4] At present, Fermilab UNIX farms comprise approximately 100 IBM RS6000 workstations (a) (upper) and 100 SGI workstations (b) (lower). Ethernet segments typically connect from 12 to 21 machines together. Each segments contains an I/O node (with SCSI-connected disk and tape) and multiple worker nodes (containing only small local disk). User jobs can utilize parts of segments (sharing I/O resources) full segments, or they can share nodes from more than one segment.

[5] Farm usage grew rapidly during the period from January 1991 to November 1992. The unit used here, VUP-months, is roughly equivalent to MIP-months, described in the text. The increase in delivered computing is the result of increases in the size of the farms and better utilization due to improved software and hardware configurations.

Figure 1a

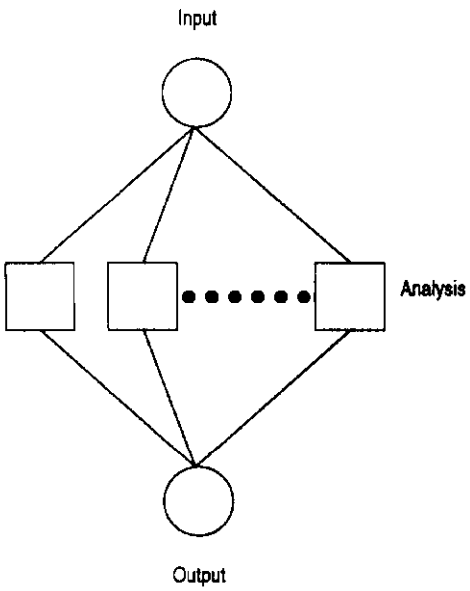

Figure 1b

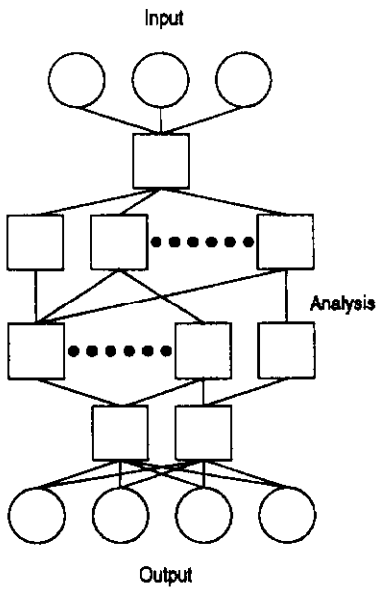

Figure 1c

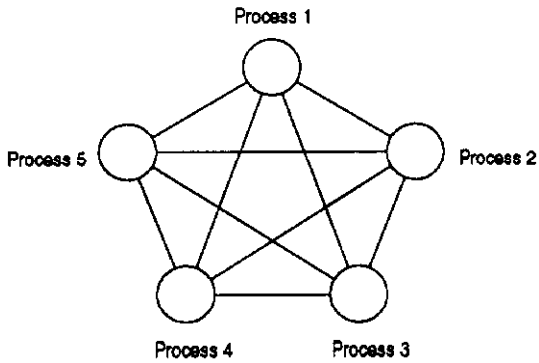

Figure 1

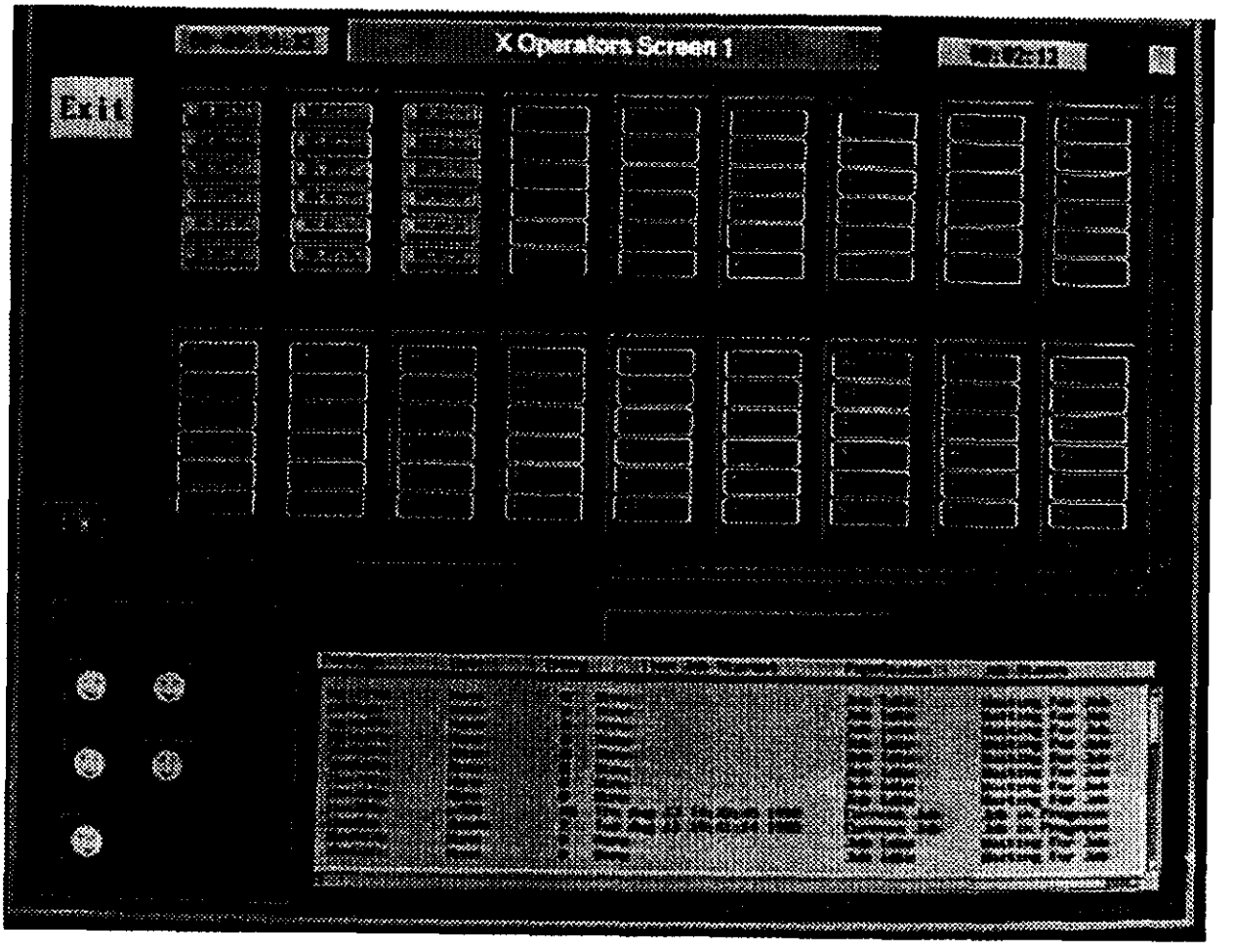

Figure 2

Figure 3

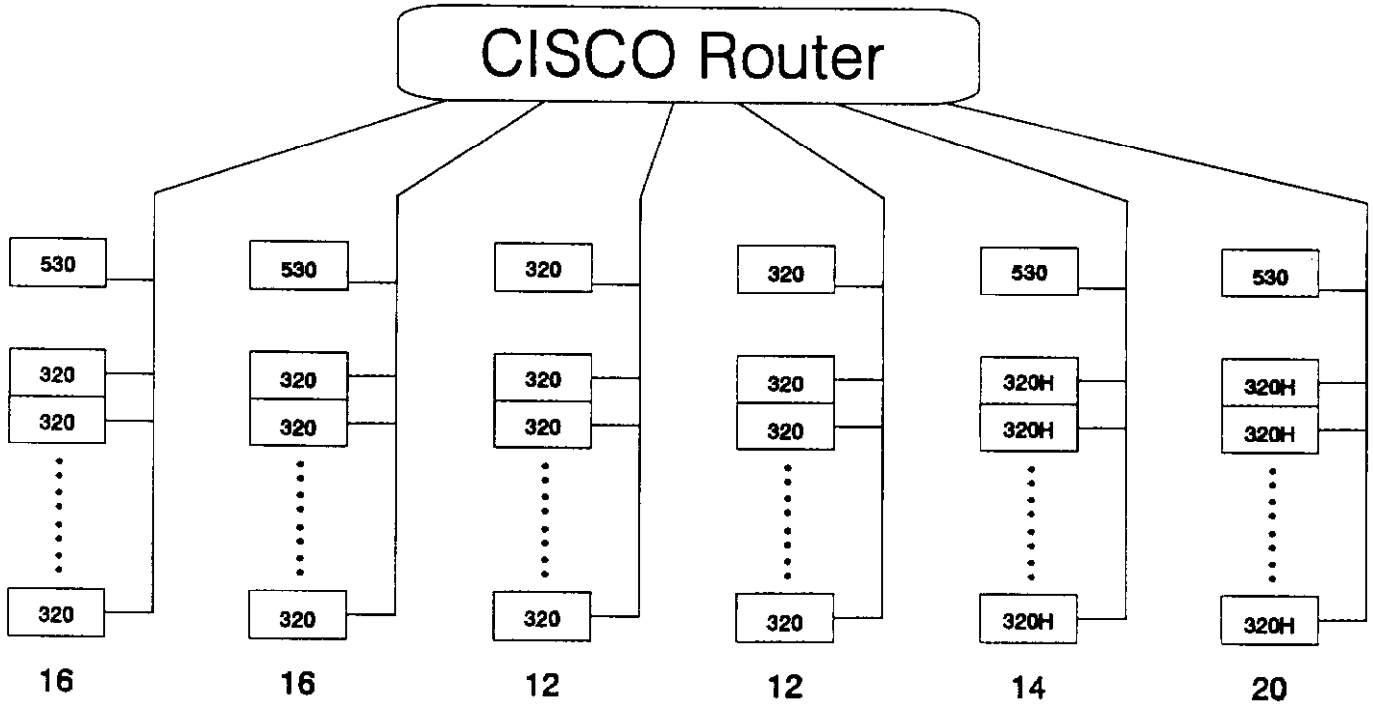

Figure 4a

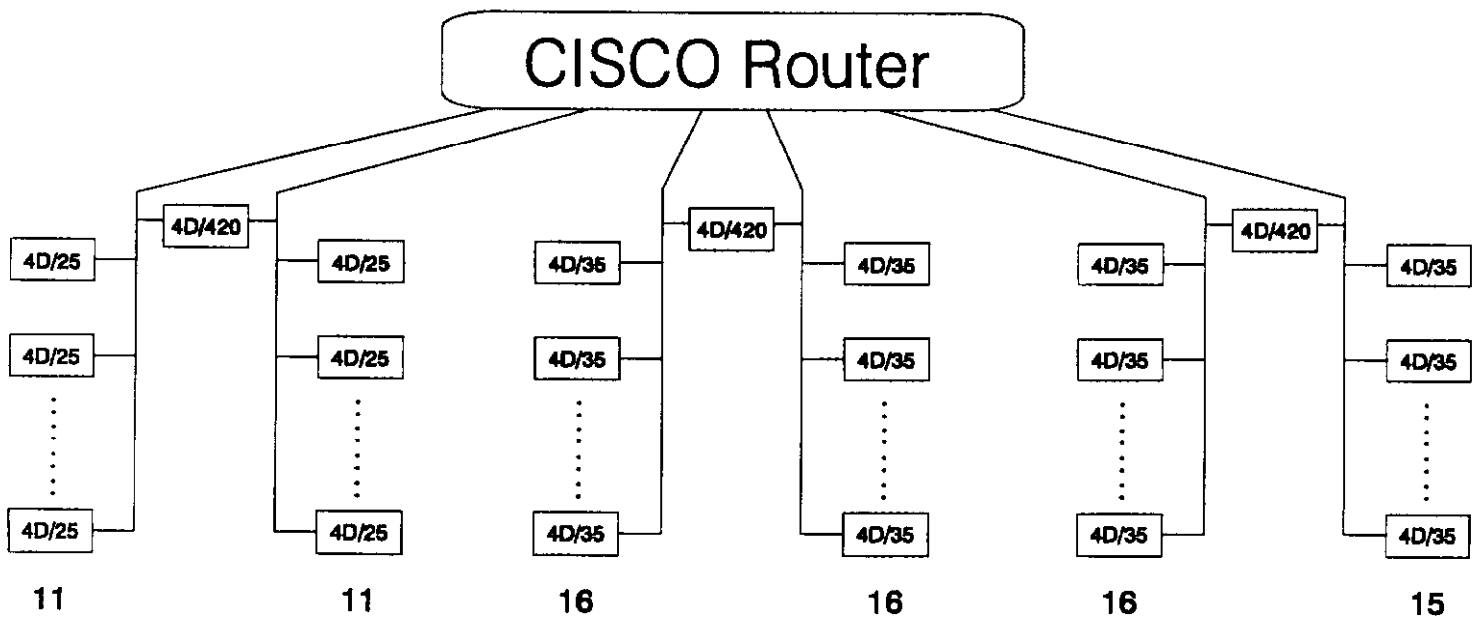

Figure 4b

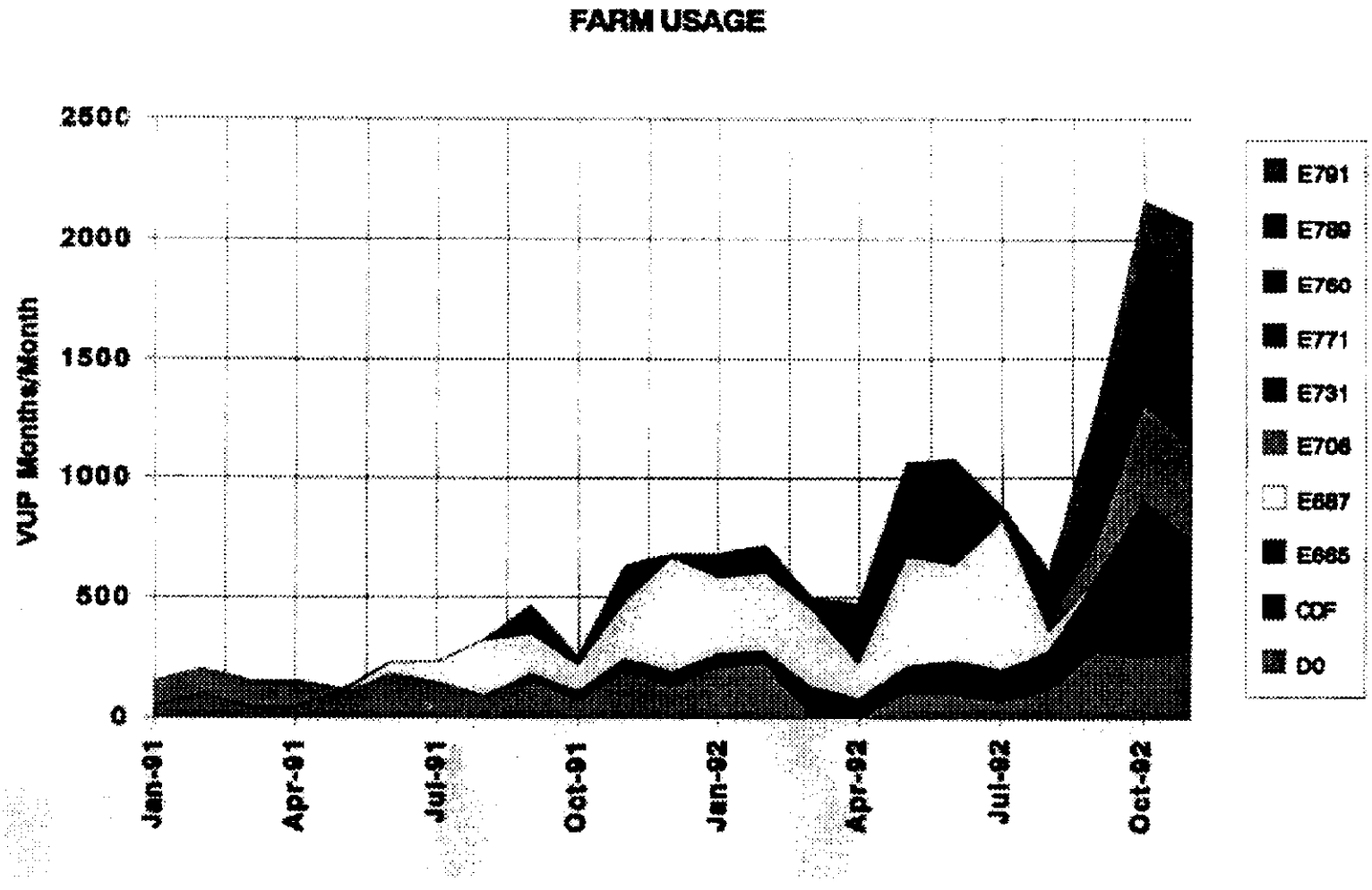
FARM USAGE
VUP Months/Month
2500
2000
1500
1000
500
0
Jan-91
Apr-91
Jul-91
Oct-91
Jan-92
Apr-92
Jul-92
Oct-92
E791
E780
E760
E771
E731
E706
E687
E685
CDF
DO

Figure 5